

A generalized Multiple-try Metropolis version of the Reversible Jump algorithm

Silvia Pandolfi^{*†}, Francesco Bartolucci^{*} and Nial Friel[‡]

June 4, 2010

Abstract

The Reversible Jump (RJ) algorithm (Green, 1995) is one of the most used Markov chain Monte Carlo algorithms for Bayesian estimation and model selection. We propose a generalized Multiple-try version of this algorithm which is based on drawing several proposals at each step and randomly choosing one of them on the basis of weights (selection probabilities) that may be arbitrary chosen. Along the same lines as in Pandolfi et al. (2010), we exploit, among the possible choices, a method based on selection probabilities depending on a quadratic approximation of the posterior distribution. Moreover, we extend these results by showing the implementation of an efficient transdimensional algorithm for challenging model selection problems. The resulting algorithm leads to a gain of efficiency with respect to the RJ algorithm also in terms of computational effort. We illustrate the approach by real examples involving a logistic regression model and a latent class model.

^{*}Department of Economics, Finance and Statistics, University of Perugia, IT

[†]email: silvia.pandolfi@stat.unipg.it

[‡]School of Mathematical Sciences, University College Dublin, IE

1 Introduction

Markov chain Monte Carlo (MCMC) methods are increasing their relevance for statistical inference, in particular Bayesian inference. In variable dimension problems, that mainly arise in the context of Bayesian model selection, a well-known approach is the Reversible Jump (RJ) algorithm proposed by Green (1995). The algorithm uses the Metropolis Hastings (MH) paradigm (Metropolis et al., 1953; Hastings, 1970) in order to generate a reversible Markov chain which jumps between models with different parameter space dimension. These jumps are achieved by proposing a move to a different model, and accepting it with appropriate probability in order to ensure that the chain has the required stationary distribution. The algorithm presents some potential drawbacks that limit its applicability. Ideally, the proposed moves are designed so that the different models are adequately explored. However, the efficient construction of these moves is generally difficult since there is no natural way to choose jump proposals (see, among others, Green, 2003).

Several approaches have been proposed in order to automate the RJ algorithm and improve its efficiency. In particular, Green and Mira (2001) exploited the delayed rejection method which is based on a modified between-model move, conditional on the rejection of an initial trial. Moreover, Brooks et al. (2003) proposed two main classes of methods. The first class is based on the efficient parametrization of the proposal density, imposing various constraints on the acceptance ratio. The second class, instead, generalizes the RJ algorithm by exploiting the so-called saturated space approach. The idea is to augment the state space with auxiliary variables (to ensure that all models share the same dimension as the largest model) and then to induce temporal memory and possible dependency in these variables. This allows the chain to have same memory of the states visited in other models, increasing the efficiency

of the proposals. Another approach to improve the efficiency of the RJ algorithm was proposed by Al-Awadhi et al. (2004), who used a secondary Markov chain to modify an initial proposal, allowing it to move from a low probability region of the new space towards a mode, before comparing it with the starting state. Fan et al. (2009) proposed the use of a convenient marginal path sampling density estimator to construct between-model proposal distributions. Finally, the idea of a fully automated RJ sampler was considered in Green and Hastie (2009), to which we also refer for a detailed review of the main methodological extensions of the RJ algorithm. Another interesting approach was proposed by Bartolucci et al. (2006), whose aim was not to improve the efficiency of the RJ algorithm but just to make use of its output, in order to construct a class of more efficient estimators of the Bayes factor.

In this paper, we extend the results illustrated in Pandolfi et al. (2010) in which a generalization of the Multiple-try Metropolis (MTM) algorithm of Liu et al. (2000) is proposed in the context of Bayesian estimation and Bayesian model choice. In particular we develop their idea of applying an MTM strategy to improve the RJ algorithm in the context of Bayesian model choice, where the dimensionality of the parameter space is also part of the model uncertainty.

In general, the MTM algorithm represents an extension of the MH algorithm consisting of proposing, at each step, a certain number of trials and then selecting one of them with a suitable probability. The selection probabilities of each proposed value are constrained so to attain the *detailed balance condition*. In particular, Liu et al. (2000) propose a rule to choose these probabilities so that they are proportional to the product of the target, the proposal, and a λ -function which is nonnegative and symmetric. These constraints lead to an algorithm that could be not so efficient as expected, because it becomes computationally intensive, especially when the number of trials is large.

The generalization of the MTM scheme proposed by Pandolfi et al. (2010), hereafter indicated by GMTM, defines the selection probabilities in a more general way. Under this approach, minimal constraints are required to attain the detailed balance condition. In principle, any mathematical function giving valid probabilities may be adopted to select among the proposed trials. In any case, the adopted function is crucial for the efficiency in the estimation of target distribution.

In the Bayesian model choice context, the GMTM extension of the RJ algorithm represents a rather natural way to overcome some of the typical problems of this algorithm. In particular, among the possible ways to compute the selection probabilities, we suggest a method based on a quadratic approximation of the target distribution that may lead to a considerable gain of efficiency. Moreover, we show that, when it is not possible to easily compute this quadratic approximation, the generalized version may again lead to an efficient algorithm, provided that the weights may be simply computed.

The paper is structured as follows. In Section 2 we review the MH algorithm and the RJ algorithm and we introduce the basic concept of GMTM algorithm for Bayesian estimation. In Section 3 we outline the MTM version of the RJ algorithm with a discussion on some convenient choices of the selection probabilities. The proposed approach is illustrated in Section 4 by some applications. Section 5 concludes.

2 Preliminaries

We first introduce the basic notation for the MH and RJ algorithms and we briefly review the GMTM method.

2.1 Metropolis Hastings and Reversible Jump algorithms

The MH algorithm, proposed by Metropolis et al. (1953) and modified by Hastings (1970), is one of the best known MCMC method that can be used to generate a random sample from a target distribution, from which direct sampling is difficult. It is well known that the basic idea of this algorithm is to construct an ergodic Markov chain in the state space of \mathbf{x} that has $\pi(\mathbf{x})$ as its stationary distribution.

A potential problem with the MH method is that the resulting samples are often highly correlated. Therefore the estimates resulting from these samples tend to have high variance. Moreover, the Metropolis-type local moves often lead to slow converging algorithms that are easily trapped in a local mode. Another challenge is the choice of an efficient proposal step. In fact, it is often the case that a small step-size in the proposal transition will result in slow convergence of the corresponding Markov chain, whereas a large step-size will result in very low acceptance rate (Liu, 2001).

In the Bayesian model choice context, the MH algorithm was extended by Green (1995) so as to allow the simulation of the posterior distribution on space of varying dimensions. It results the RJ algorithm.

Let $\{\mathcal{M}_1, \dots, \mathcal{M}_M\}$ denote the set of available models and, for model \mathcal{M}_m , let Θ_m be the parameter space, whose elements are denoted by $\boldsymbol{\theta}_m$. Also let $f(\mathbf{y}|m, \boldsymbol{\theta}_m)$ be the likelihood for an observed sample \mathbf{y} , $p(\boldsymbol{\theta}_m|m)$ be the prior distribution of the parameters and let $p(m)$ be the prior probability of model \mathcal{M}_m .

In simulating from the target distribution, a sampler must move both within and between models. Moreover, the move from the current state of Markov chain $(m, \boldsymbol{\theta}_m)$ to a new state has to be performed so as to ensure that the detailed balance condition holds. The solution proposed by Green (1995) is to introduce a set of auxiliary variables such that all states of the chain have the same dimension. For instance, a jump between models \mathcal{M}_i and \mathcal{M}_j is achieved supplementing each of the

parameter space Θ_i and Θ_j with artificial spaces in order to create a bijection between them and to impose a dimension matching condition; see also Brooks et al. (2003). In particular, let $(m, \boldsymbol{\theta}_m)$ be the current state of Markov chain, where $\boldsymbol{\theta}_m$ has dimension $d(\boldsymbol{\theta}_m)$, the algorithm performs the following steps:

Step 1: Select a new candidate model $\mathcal{M}_{m'}$ with probability $j(m, m')$.

Step 2: Generate \mathbf{u} (which can be of lower dimension than $\boldsymbol{\theta}_m$) from a specified proposal density $T(\boldsymbol{\theta}_m, \cdot)$.

Step 3: Set $(\boldsymbol{\theta}'_{m'}, \mathbf{u}') = g_{m,m'}(\boldsymbol{\theta}_m, \mathbf{u})$ where $g_{m,m'}$ is a specified invertible function. Hence $d(\boldsymbol{\theta}_m) + d(\mathbf{u}) = d(\boldsymbol{\theta}'_{m'}) + d(\mathbf{u}')$.

Step 4: The acceptance probability of the new model is:

$$\alpha = \min \left\{ 1, \frac{f(\mathbf{y}|m', \boldsymbol{\theta}'_{m'}) p(\boldsymbol{\theta}'_{m'}|m') p(m') j(m', m) T(\boldsymbol{\theta}'_{m'}, \mathbf{u}')}{f(\mathbf{y}|m, \boldsymbol{\theta}_m) p(\boldsymbol{\theta}_m|m) p(m) j(m, m') T(\boldsymbol{\theta}_m, \mathbf{u})} \left| \frac{\partial g_{m,m'}(\boldsymbol{\theta}_m, \mathbf{u})}{\partial(\boldsymbol{\theta}_m, \mathbf{u})} \right| \right\},$$

where the last term is the Jacobian of the transformation from the current value of the parameter to the new value.

The main difficulty in the implementation of the RJ algorithm regards the construction of an efficient proposal that jumps between models. Inefficient proposal mechanisms, could in fact result in Markov chains that are slow in exploring the state space and in converging to the stationary distribution. Moreover, the Markov chain will have higher autocorrelation and higher asymptotic variance with respect to Monte Carlo estimators. Generally, in order to ensure efficient proposal steps, the proposed new state should have similar posterior support to the existing state (Green and Hastie, 2009). Moreover, pilot run could be convenient to tune different aspects of the proposal mechanism.

In addition to the RJ algorithm, several alternative MCMC approaches have been proposed in Bayesian model and variable selection context. These methods are based on the estimation of the posterior probabilities of the available models or on the estimation of marginal likelihoods; for a review see Han and Carlin (2000), Dellaportas et al. (2002) and Green (2003).

The RJ algorithm belongs to the first class of methods, which also includes the product space search of Carlin and Chib (1995), the Metropolized Carlin and Chib method of Dellaportas et al. (2002) and the related composite model space approach of Godsill (2001). In particular, the Carlin and Chib (1995) scheme works in a product space framework for all possible model parameters and model indicator, so as the space has constant dimensionality and a Gibbs sampler can be used to generate samples from the posterior distribution. Dellaportas et al. (2002) proposed instead a hybrid Gibb/Metropolis strategy in which the model selection step is not based on the full conditional, but on a proposal for a move from the current model to the new model, followed by acceptance or rejection of this proposal. The composite model space method of Godsill (2001) is an RJ algorithm that takes advantage of “common” parameters in the context of variable selection. The marginal likelihood approach was proposed by Chib (1995) and extended by Chib and Jeliazkov (2001). This method is based on the estimation of the marginal likelihood of any available model from the output of the MCMC algorithm. Moreover, a powerful method for estimating the Bayes factor has been introduced by Meng and Wong (1996) on the basis of the bridge sampling identity.

The RJ algorithm has also been applied to the Bayesian analysis of data from a finite mixture distribution with an unknown number of components (Richardson and Green, 1997). This approach is based on a series of transdimensional moves (i.e. split-combine and birth-death moves) that allow the joint estimation of the parameters and

the number of components. In particular, the posterior probability of the number of components is proportional to the number of visits of the corresponding model. An interesting alternative method has been proposed by Stephens (2000) in which a continuous time version of the RJ algorithm is proposed for the analysis of finite mixture models. This approach consists of performing a birth or a death move with a certain probability. The chosen move is always accepted and the time to the next move is simulated from a suitable distribution. The posterior probability of a certain number of components is then computed as a quantity proportional to the overall permanence time in the corresponding model.

2.2 Generalized Multiple-try method

The Generalized MTM algorithm (GMTM) introduced by Pandolfi et al. (2010) use selection probabilities of the trials proposed proportional to a given *weighting function* $w^*(\mathbf{y}, \mathbf{x})$ that can be easily computed, so as to increase the number of trials without loss of efficiency.

2.2.1 The algorithm

Let $w^*(\mathbf{y}, \mathbf{x})$ be an arbitrary weighting function satisfying $w^*(\mathbf{y}, \mathbf{x}) > 0$. Let \mathbf{x}_t be the current state of Markov chain at iteration t . The GMTM algorithm performs the following step:

Step 1: Draw k trials $\mathbf{y}_1, \dots, \mathbf{y}_k$ from a proposal distribution $T(\mathbf{x}_t, \cdot)$.

Step 2: Select a point \mathbf{y} from the set $\{\mathbf{y}_1, \dots, \mathbf{y}_k\}$ with probability given by

$$p_{\mathbf{y}} = \frac{w^*(\mathbf{y}, \mathbf{x}_t)}{\sum_{j=1}^k w^*(\mathbf{y}_j, \mathbf{x}_t)}.$$

Step 3: Draw realizations $\mathbf{x}_1^*, \dots, \mathbf{x}_{k-1}^*$ from the distribution $T(\mathbf{y}, \cdot)$ and set $\mathbf{x}_k^* = \mathbf{x}_t$.

Step 4: Define $p_{\mathbf{x}_t} = \frac{w^*(\mathbf{x}_t, \mathbf{y})}{\sum_{j=1}^k w^*(\mathbf{x}_j^*, \mathbf{y})}$.

Step 5: The transition from \mathbf{x}_t to $\mathbf{x}_{t+1} = \mathbf{y}$ is accepted with probability

$$\alpha = \min \left\{ 1, \frac{\pi(\mathbf{y})T(\mathbf{y}, \mathbf{x}_t)p_{\mathbf{x}_t}}{\pi(\mathbf{x}_t)T(\mathbf{x}_t, \mathbf{y})p_{\mathbf{y}}} \right\}. \quad (1)$$

The MTM algorithm of Liu et al. (2000) can be viewed as a special case of the GMTM algorithm. In particular:

1. If $w^*(\mathbf{y}_j, \mathbf{x}_t) = \pi(\mathbf{y}_j)T(\mathbf{y}_j, \mathbf{x}_t)$, the algorithm corresponds to the MTM-I scheme with $\lambda(\mathbf{y}_j, \mathbf{x}_t) = 1$.
2. Similarly, if $w^*(\mathbf{y}_j, \mathbf{x}_t) = \frac{\pi(\mathbf{y}_j)}{T(\mathbf{x}_t, \mathbf{y}_j)}$, the corresponding MTM algorithm is based on $\lambda(\mathbf{y}_j, \mathbf{x}_t) = \left\{ T(\mathbf{y}_j, \mathbf{x}_t)T(\mathbf{x}_t, \mathbf{y}_j) \right\}^{-1}$. We term this scheme as MTM-inv.
3. If $\pi(\mathbf{y}_j)$ is replaced with its quadratic approximation, the GMTM considered in Pandolfi et al. (2010) results. We term this scheme as GMTM-quad.

Our main interest is to explore situations where the weighting function is easy to compute so as to increase the efficiency of the algorithm.

3 Multiple-try version of the RJ algorithm

As Pandolfi et al. (2010) already argued, the GMTM strategy may be applied to the RJ algorithm with the aim of developing a simultaneous inference on both model and parameter space. The Generalized MTM RJ (GMTRJ) algorithm allows us to face some of the typical drawbacks of the RJ algorithm, first of all the necessity of an accurate tuning of the jump proposals in order to promote mixing among models. The extension consists of proposing at each step a fixed number of moves, so as to improve the performance of the algorithm and to increase the efficiency from a

Bayesian model selection perspective. It is possible to consider this strategy like an automatic version of the RJ algorithm, that allows us to choose the more efficient jump proposals on the basis of selection probabilities suitably computed.

3.1 The algorithm

Suppose the Markov chain currently visits model \mathcal{M}_i with parameters $\boldsymbol{\theta}_i$ and let $w^*(\boldsymbol{\theta}_{a_j}, \boldsymbol{\theta}_i)$ be the weighting function satisfying $w^*(\boldsymbol{\theta}_{a_j}, \boldsymbol{\theta}_i) > 0$. The proposed strategy (GMTRJ) is based on the following:

Step 1: Choose a subset of models $\mathcal{M}_{\mathcal{A}} = \{M_s : s \in \mathcal{A}\}$ for some index set $\mathcal{A} = \{a_1, \dots, a_k\}$ of size k from which to propose trials.

Step 2: Draw parameters $\boldsymbol{\theta}_{a_1}, \dots, \boldsymbol{\theta}_{a_k}$ of models $\mathcal{M}_{a_1}, \dots, \mathcal{M}_{a_k}$, respectively, with proposal density $T(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{a_1}), \dots, T(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{a_k})$.

Step 3: Choose $\boldsymbol{\theta}_{a_j}$ from $\{\boldsymbol{\theta}_{a_1}, \dots, \boldsymbol{\theta}_{a_k}\}$ with probability given by

$$p_{\boldsymbol{\theta}_{a_j}} = \frac{w^*(\boldsymbol{\theta}_{a_j}, \boldsymbol{\theta}_i)}{\sum_{h=1}^k w^*(\boldsymbol{\theta}_{a_h}, \boldsymbol{\theta}_i)}.$$

Step 4: Choose a subset of models $\mathcal{M}_{\mathcal{B}} = \{m_t : t \in \mathcal{B}\}$ for some index set $\mathcal{B} = \{b_1, \dots, b_k\}$ where $b_k = i$.

Step 5: Draw parameters $\boldsymbol{\theta}_{b_1}, \dots, \boldsymbol{\theta}_{b_{k-1}}$ of $\mathcal{M}_{b_1}, \dots, \mathcal{M}_{b_{k-1}}$, with proposal density $T(\boldsymbol{\theta}_{a_j}, \boldsymbol{\theta}_{b_1}), \dots, T(\boldsymbol{\theta}_{a_j}, \boldsymbol{\theta}_{b_{k-1}})$ and set $\boldsymbol{\theta}_{b_k} = \boldsymbol{\theta}_i$.

Step 6: Define $p_{\boldsymbol{\theta}_i} = \frac{w^*(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{a_j})}{\sum_{h=1}^k w^*(\boldsymbol{\theta}_{b_h}, \boldsymbol{\theta}_{a_j})}$.

Step 7: Accept the move from $\boldsymbol{\theta}_i$ to $\boldsymbol{\theta}_{a_j}$ with probability

$$\alpha = \min \left\{ 1, \frac{\pi(\boldsymbol{\theta}_{a_j})T(\boldsymbol{\theta}_{a_j}, \boldsymbol{\theta}_i)p_{\boldsymbol{\theta}_i}}{\pi(\boldsymbol{\theta}_i)T(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{a_j})p_{\boldsymbol{\theta}_{a_j}}} |J(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{a_j})| \right\}.$$

where $|J(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{a_j})|$ is the Jacobian of the transformation from the current value of the parameters to the new value.

It is possible to prove that the GMTRJ algorithm satisfies detailed balance condition; see Theorem 3.1 in the following section.

3.2 Prove of detailed balance condition

As is common in the MCMC approach, the generated chain has to be reversible and to satisfied the *detailed balance condition*, i.e.

$$\pi(\mathbf{y})P(\mathbf{y}, \mathbf{x}) = \pi(\mathbf{x})P(\mathbf{x}, \mathbf{y}), \quad (2)$$

for every (\mathbf{x}, \mathbf{y}) , where $P(\mathbf{y}, \mathbf{x})$ is the transition kernel from \mathbf{y} to \mathbf{x} . This condition defines a situation of equilibrium in the Markov chain, namely that the probability of being in \mathbf{x} and moving to \mathbf{y} is the same as the probability of being in \mathbf{y} and moving back to \mathbf{x} (see Robert and Casella (2004) for more details).

In the following Theorem we demonstrate that the detailed balance condition holds in the generalized version of the RJ algorithm.

Theorem 3.1. *The GMTRJ algorithm satisfies detailed balance.*

Proof. The GMTRJ algorithm involves transitions to states of variable dimension, and consequently the detailed balance condition is now written as

$$\pi(\boldsymbol{\theta}_i)P(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{a_j}) = \pi(\boldsymbol{\theta}_{a_j})P(\boldsymbol{\theta}_{a_j}, \boldsymbol{\theta}_i)|J(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{a_j})|.$$

where, as above, $\boldsymbol{\theta}_i$ represents the current value of the parameter vector and $\boldsymbol{\theta}_{a_j}$ is one of the new parameters proposed with $j = 1, \dots, k$.

Suppose that $\theta_i \neq \theta_{a_j}$, noting that $\theta_{a_1}, \dots, \theta_{a_k}$ are exchangeable, it holds that

$$\begin{aligned}
\pi(\theta_i) P(\theta_i, \theta_{a_k}) &= \\
&= k \pi(\theta_i) T(\theta_i, \theta_{a_k}) p_{\theta_{a_k}} \int \dots \int T(\theta_i, \theta_{a_1}) \dots T(\theta_i, \theta_{a_{k-1}}) \\
&\quad \min \left\{ 1, \frac{\pi(\theta_{a_k}) T(\theta_{a_k}, \theta_i) p_{\theta_i}}{\pi(\theta_i) T(\theta_i, \theta_{a_k}) p_{\theta_{a_k}}} |J(\theta_i, \theta_{a_k})| \right\} \\
&\quad T(\theta_{a_k}, \theta_{b_1}) \dots T(\theta_{a_k}, \theta_{b_{k-1}}) d\theta_{a_1} \dots d\theta_{a_{k-1}} d\theta_{b_1} \dots d\theta_{b_{k-1}} \\
&= k \int \dots \int T(\theta_i, \theta_{a_1}) \dots T(\theta_i, \theta_{a_{k-1}}) \\
&\quad \min \{ \pi(\theta_i) T(\theta_i, \theta_{a_k}) p_{\theta_{a_k}}, \pi(\theta_{a_k}) T(\theta_{a_k}, \theta_i) p_{\theta_i} |J(\theta_i, \theta_{a_k})| \} \\
&\quad T(\theta_{a_k}, \theta_{b_1}) \dots T(\theta_{a_k}, \theta_{b_{k-1}}) d\theta_{a_1} \dots d\theta_{a_{k-1}} d\theta_{b_1} \dots d\theta_{b_{k-1}} \\
&= k \pi(\theta_{a_k}) T(\theta_{a_k}, \theta_i) p_{\theta_i} |J(\theta_i, \theta_{a_k})| \\
&\quad \int \dots \int T(\theta_i, \theta_{a_1}) \dots T(\theta_i, \theta_{a_{k-1}}) \\
&\quad \min \left\{ 1, \frac{\pi(\theta_i) T(\theta_i, \theta_{a_k}) p_{\theta_{a_k}}}{\pi(\theta_{a_k}) T(\theta_{a_k}, \theta_i) p_{\theta_i}} \frac{1}{|J(\theta_i, \theta_{a_k})|} \right\} \\
&\quad T(\theta_{a_k}, \theta_{b_1}) \dots T(\theta_{a_k}, \theta_{b_{k-1}}) d\theta_{a_1} \dots d\theta_{a_{k-1}} d\theta_{b_1} \dots d\theta_{b_{k-1}} \\
&= \pi(\theta_{a_k}) P(\theta_{a_k}, \theta_i) |J(\theta_i, \theta_{a_k})|,
\end{aligned}$$

as required. Note that $|J(\theta_{a_k}, \theta_i)| = 1/|J(\theta_i, \theta_{a_k})|$. □

3.3 Choice of the weighting function

The choice of the weighting function $w^*(\theta_{a_j}, \theta_i)$ is crucial for the efficient construction of the jump proposal. In fact, with an useful choice of this weight, it may be possible to construct an algorithm that is easy to implement and that allows us to reach a good acceptance rate, together with a gain of efficiency.

At this aim, we consider some special cases of the GMTRJ algorithm:

1. if $w^*(\theta_{a_h}, \theta_i) = \pi(\theta_{a_h}) T(\theta_{a_h}, \theta_i)$, then we have the MTRJ-I scheme that corre-

sponds to the MTM-I scheme in Bayesian estimation framework;

2. if $w^*(\boldsymbol{\theta}_{a_h}, \boldsymbol{\theta}_i) = \frac{\pi(\boldsymbol{\theta}_{a_h})}{T(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{a_h})}$, then we have the MTRJ-inv scheme that corresponds to the MTM-inv scheme;
3. if $w^*(\boldsymbol{\theta}_{a_h}, \boldsymbol{\theta}_i) = \frac{\pi^*(\boldsymbol{\theta}_{a_h})}{T(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{a_h})}$, where $\pi^*(\boldsymbol{\theta}_{a_h})$ is the quadratic approximation of the target distribution, then the GMTRJ-quad scheme results;
4. If it is not possible to simply derive the quadratic approximation of the target distribution it is always possible to find a suitable function that allows us to simplify the computations.

In particular, in the GMTRJ-quad algorithm the quadratic approximation of the target distribution is given by

$$\begin{aligned}\pi^*(\boldsymbol{\theta}_{a_h}) &= \pi(\boldsymbol{\theta}_i)A(\boldsymbol{\theta}_{a_h}, \boldsymbol{\theta}_i) \\ A(\boldsymbol{\theta}_{a_h}, \boldsymbol{\theta}_i) &= \exp \left[\boldsymbol{s}(\boldsymbol{\theta}_i)'(\boldsymbol{\theta}_{a_h} - \boldsymbol{\theta}_i) + \frac{1}{2}(\boldsymbol{\theta}_{a_h} - \boldsymbol{\theta}_i)' \boldsymbol{D}(\boldsymbol{\theta}_i)(\boldsymbol{\theta}_{a_h} - \boldsymbol{\theta}_i) \right],\end{aligned}$$

where $\boldsymbol{s}(\boldsymbol{\theta})$ and $\boldsymbol{D}(\boldsymbol{\theta})$ are, respectively, the first and second derivatives of $\log \pi(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$. Then, in the computation of the selection probabilities, after some simple algebra, we find an expression that does not require to compute the target distribution for each proposed value, saving much computing time.

An example in which is not feasible to compute this quadratic approximation (considered in the fourth case above), is instead the Bayesian model selection of the number of unknown classes in a latent class model (example outlined in the following section). In this case, the weighting function may be computed as a quantity proportional to the *incomplete likelihood* based on the manifest distribution of the observable data. The estimation algorithm is, in fact, based on the concept of *complete data*, which consist of the response configuration (incomplete data) and the configuration

of the latent variable (or allocation variable) z_i that denotes the subpopulation in which the i -th individual belongs to. Thus the incomplete likelihood does not include the allocation variable z_i and the weighting function can be easily computed while still being the proposal efficient. We term this version as GMTRJ-man.

4 Empirical illustrations

We tested our approach in two different examples about Bayesian model selection. The first one concerns the selection of the covariates in a logistic regression model, the second one is about the choice of the number of components of a latent class model. Both of the examples have already been illustrated in Pandolfi et al. (2010), in which the former has been analyzed in some detail while the latter has been only briefly explained.

4.1 Logistic regression analysis

We considered a logistic regression model for the number of survivals, Y , in a sample of 79 subjects suffering a certain illness. The patient condition, A , and the received treatment, B , are the explanatory factors. See Dellaportas et al. (2002) for details. With respect to Pandolfi et al. (2010), we only focused on the Bayesian model choice problem, reporting more extended results.

We considered five possible models: \mathcal{M}_1 (intercept); \mathcal{M}_2 (intercept + A); \mathcal{M}_3 (intercept + B); \mathcal{M}_4 (intercept + A + B); \mathcal{M}_5 (intercept + A + B + $A.B$). The last model, also termed as full model, is formulated as

$$Y_{ij} \sim \text{Bin}(n_{ij}, p_{ij}), \quad \text{logit}(p_{ij}) = \mu + \mu_i^A + \mu_j^B + \mu_{ij}^{AB}$$

where, for $i, j = 1, 2$, Y_{ij} , n_{ij} and p_{ij} are, respectively, the number of survivals, the total number of patients and the probability of survival for the patients with condition i who received treatment j . Let $\boldsymbol{\beta} = (\beta_0, \beta_1, \beta_2, \beta_3) = (\mu, \mu_2^A, \mu_2^B, \mu_{22}^{AB})$ be the parameter vector of the model. As in Dellaportas et al. (2002) we used the prior $N(0, 8)$ for any of these parameters, which by assumption are also a priori independent.

With the aim of testing the performance of the proposed approach, we compared the results of the RJ algorithm with those of the MTRJ-I, MTRJ-inv and GMTRJ-quad algorithms. The proposal distribution that we used to update the parameters and to jump from one model to another is $\boldsymbol{\beta}_{t+1} \sim N(\boldsymbol{\beta}_t, \sigma_p^2)$. As in Pandolfi et al. (2010), we evaluated the effect of the proposal distribution on the performance of the algorithms considering several different values of σ_p (0.1, 0.2, 0.5, 1, 1.5, 2, 2.5). Moreover, for the last two algorithms, we considered three different numbers of trials ($k = 10, 50, 100$). All the Markov chains were initialized from the full model with starting point $\boldsymbol{\beta} = \mathbf{0}$, and their moves were restricted to adjacent models (which increases or decreases the model dimension by 1). In the MTRJ-I, MTRJ-inv and GMTRJ algorithms, the MTM strategy is only applied in drawing the parameter values. For all the algorithms, every sweep consists of a move aimed at updating the parameters of the current model (performed through the GMTM-quad algorithm), and of a transdimensional move aimed at jumping from a model to another. Finally each Markov chain ran for 1,000,000 iterations discarding the first 50,000 as burn-in.

The output summaries are illustrated in Table 1 for $\sigma_p = 0.5$ and a number of trials $k = 10$; all of the approaches gave similar results. Figure 1 illustrates the evolution of the ergodic probabilities for the two most probable models (\mathcal{M}_2 and \mathcal{M}_4) in the first 60,000 iterations. We considered two different values of σ_p (0.5, 2) and, for the MTRJ and GMTRJ-quad algorithms, a number of trials k equal to 10.

Table 1: Posterior model probabilities for the logistic example with $\sigma_p = 0.5$ and $k = 10$

Model	RJ	MTRJ-I	MTRJ-inv	GMTRJ-quad
$\mathcal{M}_1 = \mu$	0.0050	0.0049	0.0049	0.0049
$\mathcal{M}_2 = \mu + \mu_i^A$	0.4889	0.4922	0.4930	0.4928
$\mathcal{M}_3 = \mu + \mu_j^B$	0.0111	0.0111	0.0113	0.0112
$\mathcal{M}_4 = \mu + \mu_i^A + \mu_j^B$	0.4425	0.4398	0.4391	0.4380
$\mathcal{M}_5 = \mu + \mu_i^A + \mu_j^B + \mu_{ij}^{AB}$	0.0524	0.0519	0.0516	0.0521

We can see that the RJ algorithm has slower convergence than MTRJ-I, MTRJ-inv and GMTRJ-quad algorithms, especially when the proposal is not adequate ($\sigma_p = 2$).

The efficiency of the algorithms is measured on the basis of the ratio $R = \sigma_a^2 / \sigma^2$, where σ^2 is the Monte Carlo variance of the mean estimator and σ_a^2 is the asymptotic variance of the same estimator based on the draws generated by the algorithm of interest. In particular, σ_a^2 is computed on the basis of the autocorrelation between these draws and takes into account the permanence in the same model. Limited to the case of $k = 50$ trials, and to the RJ, MTRJ-inv and GMTRJ-quad algorithms (we do not report the results of the MTRJ-I algorithm since they are quite similar to those of the MTRJ-inv algorithm), the results of the above comparison are reported in Figure 2, which shows how the value of the index R (corrected for the computing time) behaves as σ_p increases. Table 2 summarizes these results for all the values of k considered and for three different values of σ_p (0.1, 0.5, 2). In Figure 3 we also report the efficiency ratios corrected again for the computing time.

We observe that, for most of values of σ_p there is a consistent gain of efficiency of the MTRJ-inv and GMTRJ-quad algorithms with respect to RJ algorithm. This gain of efficiency corresponds to an increase of the acceptance rate, which is higher in both the MTRJ-inv and the GMTRJ-quad algorithms with respect to the RJ algorithm (see Table 3 which shows the acceptance rate for all the values of σ_p and

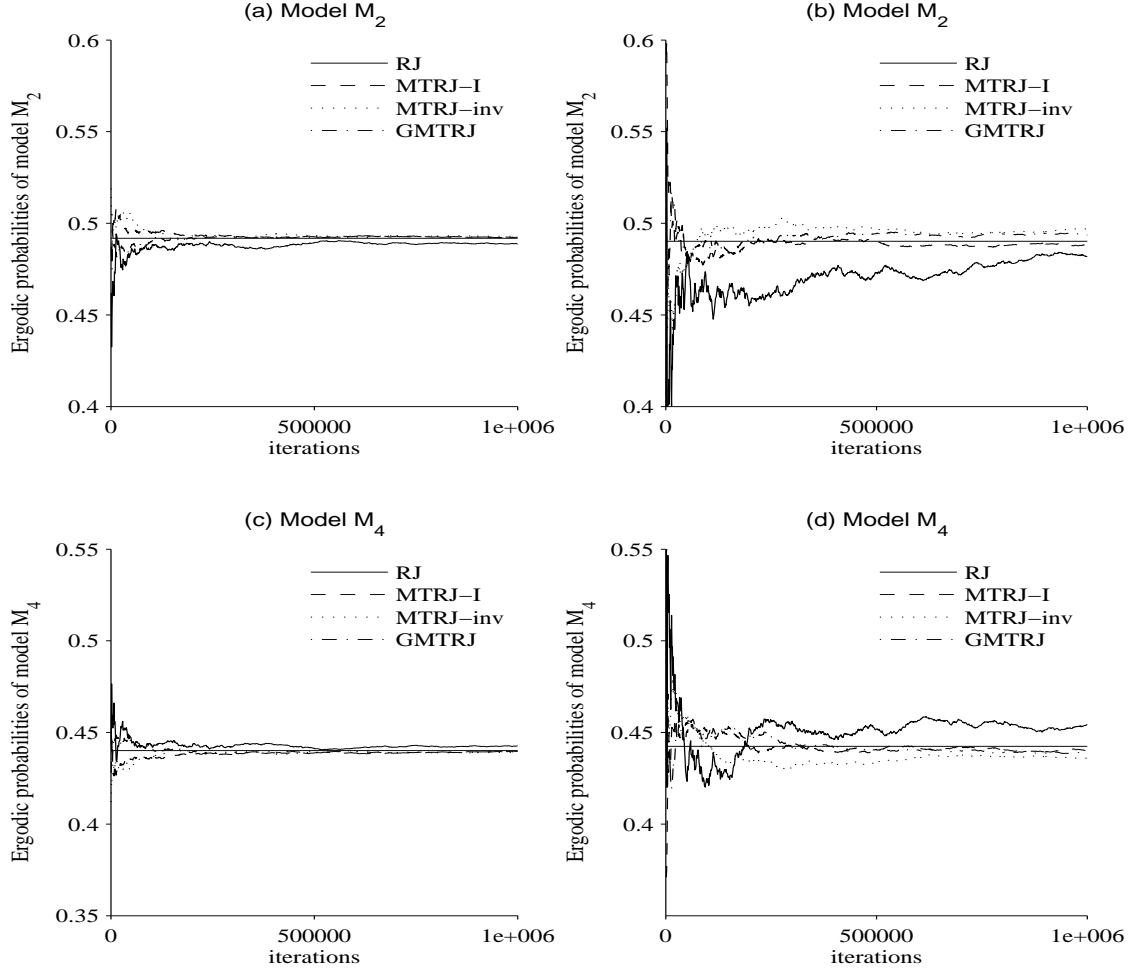


Figure 1: Ergodic posterior model probability of models \mathcal{M}_2 and \mathcal{M}_4 for the logistic regression example: (a) Model \mathcal{M}_2 , $\sigma_p = 0.5$, $k = 10$, (b) Model \mathcal{M}_2 , $\sigma_p = 2$, $k = 10$, (c) Model \mathcal{M}_4 , $\sigma_p = 0.5$, $k = 10$, (d) Model \mathcal{M}_4 , $\sigma_p = 2$, $k = 10$

k considered). It is also worth noting that, the GMTRJ-quad algorithm leads to an acceptance rate which is slightly lower than that of the MTRJ-inv algorithm, but this reduction is more than compensated by the saved computing time due to the use of the quadratic approximation. Overall, we can see that the proposed GMTRJ-quad algorithm outperforms the other two algorithms, when the computing time is properly taken into account.

Table 2: Values (divided by 1000 and adjusted for the computing time) of R for the logistic example

			\mathcal{M}_1	\mathcal{M}_2	\mathcal{M}_3	\mathcal{M}_4	\mathcal{M}_5
	RJ	$\sigma_p = 0.1$	8.84	17.92	14.06	14.05	3.95
		$\sigma_p = 0.5$	2.05	3.33	3.14	2.72	1.89
		$\sigma_p = 2$	7.57	50.05	22.12	40.43	96.36
$k = 10$	MTRJ-inv	$\sigma_p = 0.1$	8.90	18.08	12.29	14.76	4.09
		$\sigma_p = 0.5$	1.52	2.47	2.17	2.02	1.49
		$\sigma_p = 2$	2.36	13.13	5.76	11.41	19.03
	GMTRJ-quad	$\sigma_p = 0.1$	8.27	19.87	13.87	15.59	4.51
		$\sigma_p = 0.5$	1.48	2.41	1.85	1.99	1.53
		$\sigma_p = 2$	2.30	12.40	5.45	10.81	22.74
$k = 50$	MTRJ-inv	$\sigma_p = 0.1$	7.32	19.37	11.50	15.67	4.71
		$\sigma_p = 0.5$	2.02	2.86	2.58	2.38	1.82
		$\sigma_p = 2$	2.10	5.01	3.32	4.58	7.34
	GMTRJ-quad	$\sigma_p = 0.1$	5.99	13.13	9.93	10.22	4.05
		$\sigma_p = 0.5$	1.59	2.56	2.02	2.11	1.57
		$\sigma_p = 2$	1.70	4.70	2.81	3.80	6.87
$k = 100$	MTRJ-inv	$\sigma_p = 0.1$	8.72	23.66	13.86	18.94	6.44
		$\sigma_p = 0.5$	2.62	3.57	3.41	3.01	2.42
		$\sigma_p = 2$	2.83	5.18	3.71	4.40	6.09
	GMTRJ-quad	$\sigma_p = 0.1$	6.45	16.81	10.17	13.41	4.30
		$\sigma_p = 0.5$	1.80	3.04	2.40	2.49	1.90
		$\sigma_p = 2$	1.94	4.03	2.72	3.30	4.50

Table 3: Acceptance rate for the logistic example

		σ_p						
		0.1	0.2	0.5	1	1.5	2	2.5
	RJ	9.24	14.07	10.84	3.56	1.46	0.72	0.43
$k = 10$	MTRJ-inv	11.53	21.82	33.84	20.42	10.83	6.13	3.82
	GMTRJ-quad	11.44	21.72	31.19	19.01	10.24	5.86	3.68
$k = 50$	MTRJ-inv	12.89	26.27	40.71	35.81	26.28	18.31	12.82
	GMTRJ -quad	12.97	25.91	35.52	31.29	23.65	16.74	11.99
$k = 100$	MTRJ-inv	13.60	27.98	41.53	39.35	32.69	25.15	18.95
	GMTRJ-quad	13.56	27.48	35.96	33.85	28.79	22.62	17.31

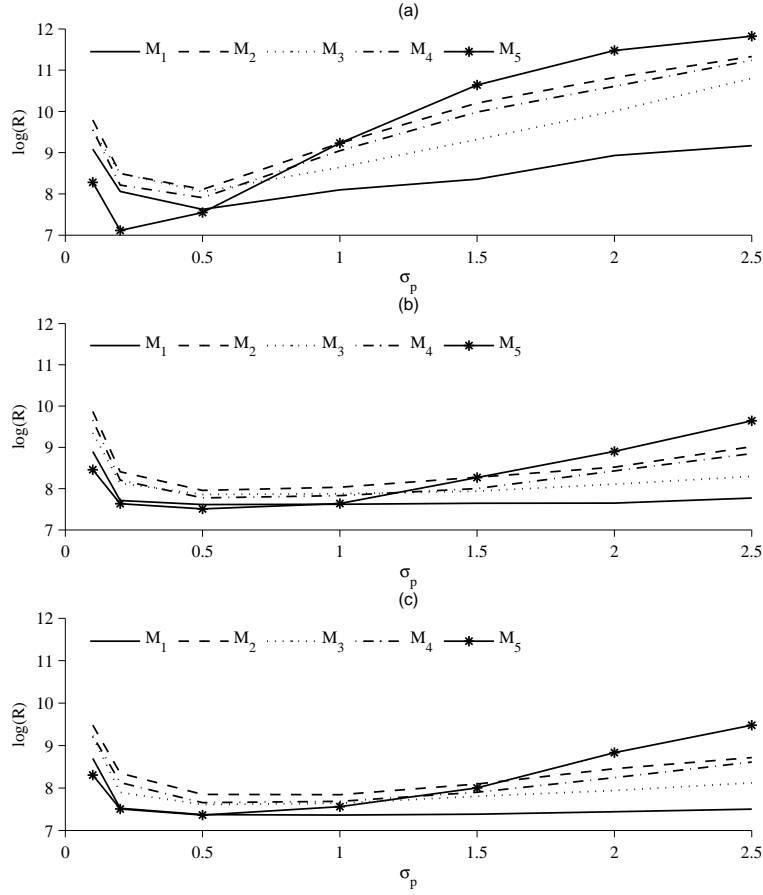


Figure 2: Values of the index R as σ_p increases with $k = 50$ (computing time taken into account): (a) RJ, (b) MTRJ-inv, (c) GMTRJ-quad

4.2 Latent class analysis

We considered the same latent class model and the same data considered by Goodman (1974), which concern the responses to four dichotomous items of a sample of 216 respondents. These items were about the personal feeling toward four situations of role conflict; then there are four response variables collected in the vector $\mathbf{Y} = (Y_1, Y_2, Y_3, Y_4)$.

Parameters of the model are the class weights π_c and the conditional probabilities of success $\lambda_{j|c}$, where $c = 1, \dots, C$, with C denoting the number of unknown classes and $j = 1, \dots, 4$. On the basis of these parameters, the probability of a response

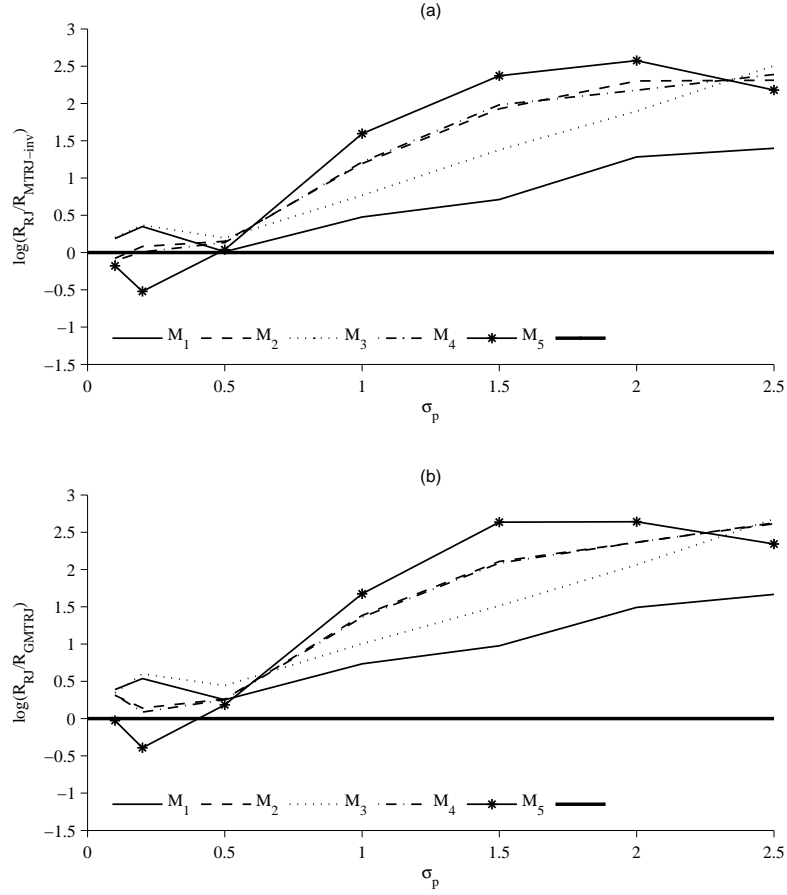


Figure 3: Efficiency ratio as σ_p increases with $k = 50$ (computing time taken into account): (a) RJ versus MTRJ-inv, (b) RJ versus GMTRJ-quad.

configuration \mathbf{y} , with elements y_j , is given by

$$P(\mathbf{y}) = \sum_{c=1}^C \pi_c \prod_{j=1}^4 \lambda_{j|c}^{y_j} (1 - \lambda_{j|c})^{1-y_j}.$$

A priori, we assumed a Dirichlet distribution for the parameter vector $(\pi_1, \dots, \pi_C) \sim D(\delta, \dots, \delta)$, with $\delta = 1$, and independent Beta distributions for the parameters $\lambda_{j|c}$. We set both the parameter of the Beta distribution equal to 1. Finally, for C we assumed a uniform distribution between 1 and C_{\max} , where C_{\max} is the maximum number of classes we assumed a priori.

The objective of the analysis is the inference about the number of classes (C),

and the parameters $\lambda_{j|c}$ and π_c . At this aim we exploited the approach of Richardson and Green (1997), who applied the RJ algorithm on the analysis of finite mixtures of normal densities with unknown number of components. On the basis of this approach, we performed a RJ strategy where the moves are restricted to models with one more or one less component. We associated to each subject in the sample an *allocation variable* z_i equal to c when subject i belongs to latent class c . The a priori distribution of each z_i depends on the class weights π_c ; see also Cappe et al. (2003). This strategy is called *completing the sample*, and (z, \mathbf{y}) is referred to as the complete data. Under this formulation the *complete data likelihood* has logarithm

$$l^*(\boldsymbol{\theta}) = \sum_z \sum_{\mathbf{y}} m_{z\mathbf{y}} \log[f(z, \mathbf{y})]$$

where $\boldsymbol{\theta}$ is the vector of all model parameters arranged in a suitable way, $m_{z\mathbf{y}}$ are the frequencies of the contingency table in which the subjects are cross-classified according to latent configuration z and response configuration \mathbf{y} , and $f(z, \mathbf{y})$ is the manifest distribution

$$f(z, \mathbf{y}) = \pi_z \prod_j \lambda_{j|z}^{y_j} (1 - \lambda_{j|z})^{(1-y_j)}.$$

We considered two different pair of dimension-changing moves: split-combine and birth-death which are performed with probability equal to 0.5 respectively. At every iteration, split-combine or birth-death moves are preceded by a Gibbs move aimed at updating the parameters of the current model, sampling from the full conditional distribution. We tackled the label switching problem by post-processing the output of the algorithm: we used relabeling-invariant priors and ran the MCMC unconstrained. Then the MCMC output was sorted on the basis of the class weights.

Suppose that the current state of the chain is $(C, \boldsymbol{\theta}_C)$. In the split-combine move,

we first make a random choice between attempting to split or combine with probability 0.5. Of course, if $C = 1$ we always propose a split move and if $C = C_{\max}$ we always propose a combine move. The split proposal consists of choosing a class c^* at random and split into two new ones, labeled c_1 and c_2 . The corresponding parameters are split as follows:

1. $\pi_{c_1} = \pi_{c^*} \times u$ and $\pi_{c_2} = \pi_{c^*} \times (1 - u)$ with $u \sim Be(\alpha, \beta)$;
2. $\lambda_{j|c_1} \sim Be(\tau \times \lambda_{j|c^*}, \tau \times (1 - \lambda_{j|c^*}))$ and $\lambda_{j|c_2} \sim Be(\tau \times \lambda_{j|c^*}, \tau \times (1 - \lambda_{j|c^*}))$, for $j = 1, \dots, 4$, where τ is a given constant that has to be tuned in order to reach an adequate acceptance rate.

When the split move is accomplished, it remains only to propose the reallocation of those observations with $z_i = c^*$ between c_1 and c_2 . The allocation is done on the basis of probabilities computed analogously to the Gibbs allocation.

In the reverse combine move a pair of classes (c_1, c_2) is picked at random and merged into a new one, c^* , as follows:

1. $\pi_{c^*} = \pi_{c_1} + \pi_{c_2}$;
2. $\lambda_{j|c^*} \sim Be(\tau \times \lambda_m, \tau \times (1 - \lambda_m))$, with $\lambda_m = (\lambda_{j|c_1} + \lambda_{j|c_2})/2$ for $j = 1, \dots, 4$.

The reallocation of the observations with $z_i = c_1$ or $z_i = c_2$ is done by setting $z_i = c^*$.

The split move is accepted with probability $\min\{1, A\}$ whereas the combine move is accepted with probability $\min\{1, A^{-1}\}$ where A can be computed as

$$\begin{aligned}
& \frac{f^*(\mathbf{y}|\boldsymbol{\theta}_{C+1})p(\boldsymbol{\theta}_{C+1})p(C+1)}{f^*(\mathbf{y}|\boldsymbol{\theta}_C)p(\boldsymbol{\theta}_C)p(C)} \times \frac{(C+1)!}{C!} \times \frac{P_c(C+1)/[(C+1)C/2]}{P_s(C)/C} \times \frac{1}{P_{alloc}} \\
& \times \frac{\prod_j p(\lambda_{j|c^*})}{2p(u) \prod_j p(\lambda_{j|c_1})p(\lambda_{j|c_2})} \times |J_{split}| \\
& = \frac{f^*(\mathbf{y}|\boldsymbol{\theta}_{k+1})p(\boldsymbol{\theta}_{k+1})}{f^*(\mathbf{y}|\boldsymbol{\theta}_k)p(\boldsymbol{\theta}_k)} \times \frac{P_c(C+1)}{P_s(C)P_{alloc}} \times \frac{\prod_j g(\lambda_{j|c^*})}{g(u) \prod_j g(\lambda_{j|c_1})g(\lambda_{j|c_2})} \times |J_{split}|,
\end{aligned} \tag{3}$$

where $f^*(\mathbf{y}|\boldsymbol{\theta}_C)$ is the complete data likelihood not in logarithmic form, and g denotes the Beta density. $P_s(C)/C$ and $P_c(C+1)/[(C+1)C/2]$ are respectively the probabilities to split a specific class out of C available ones and to combine one of $(C+1)C/2$ possible pairs of classes. The factorials and the coefficient 2 arise from combinatorial reasoning related to label switching; P_{alloc} is the probability that this particular allocation is made and $|J_{split}|$ is the Jacobian of the transformation from $\boldsymbol{\theta}_C$ to $\boldsymbol{\theta}_{C+1}$, which is equal to π_{c^*} .

In the birth-death move we first propose a birth or a death move along the same lines as above; then a birth is accomplished by generating a new empty class, i.e. a class to which no observation is allocated, denoted by c^* . To do this we draw π_{c^*} from $Be(1, C)$, where C is the current number of classes, and rescale the existing weights, so that they sum to 1, as $\pi'_c = \pi_c(1 - \pi_{c^*})$. The new parameters $\lambda_{j|c^*}$ for $j = 1, \dots, 4$ are drawn from their prior distribution.

For the death move, a random choice is made between the empty classes; the chosen class is deleted and the remaining class weights are rescaled to sum to 1. The allocation of the z_i is unaltered because the class deleted is empty.

The use of the prior distribution in proposing the new values for $\lambda_{j|c^*}$ leads to a simplification of the resulting acceptance probability, $\min\{1, A\}$, that, after some calculation, reduces to

$$\frac{\pi_{c^*}^{\delta-1}(1 - \pi_{c^*})^{n+C\delta-C}}{B(C\delta, \delta)} \times \frac{P_d(C+1)}{P_b(C)} \times \frac{(C+1)}{(C_0+1)} \times \frac{1}{g(\pi_{c^*})} \times |J_{birth}|.$$

Here, the first term is the prior ratio, and the remaining terms contain the proposal ratio; $B(\cdot, \cdot)$ is the Beta function, C_0 is the number of empty classes and the likelihood ratio is 1. The Jacobian is computed as $|J_{birth}| = (1 - \pi_{c^*})^{C-1}$. The death move is accepted with probability $\min\{1, A^{-1}\}$.

In this application, we compared the standard RJ algorithm with two different versions of the MTRJ-inv algorithm. In the first version, named MTRJ-inv-I, the MTM strategy consists of proposing, at each step, a fixed number k of different classes to split or combine or to add or delete; then the corresponding parameters are drawn from their respective proposal distributions. In the second version, named MTRJ-inv-II, we choose a class to split or combine (or to add or delete) and the MTM strategy is only applied in drawing the parameter values. Moreover, we applied the GMTRJ-man algorithm to the data, in which the selection probabilities are computed as a value proportional to the incomplete likelihood. In fact, this is a situation in which the quadratic approximation of the target distribution can not be easily computed. The incomplete likelihood does not include the allocation variables z_i , which have not to be reallocated for each proposed trial. This allows us to save much computing time. As in the MTRJ-inv algorithm, we tested two versions of the GMTRJ-man (GMTRJ-man-I and GMTRJ-man-II) using the same criterion as above.

We ran each Markov chain for 2,000,000 sweeps following a burn-in of 400,000 iterations; moreover, we set $C_{max} = 20$, $\alpha = \beta = 2$ and $\tau = 10$. For both the versions of the MTRJ-inv and GMTRJ-man algorithms, we also considered two different number of trials ($k = 5, 10$).

Table 4 shows the estimated posterior probabilities of each class for all the algorithms, using $k = 10$. We can see that all the algorithms give quite similar posterior probabilities of the number of classes; the two most probable models are the model with two classes and the model with three classes. Table 5 illustrates the proportion of moves accepted. The plot of the first 40,000 values of C after the burn-in is given in Figure 4 while, in order to check the stationarity, Figure 5 illustrates the plot of the cumulative occupancy fractions for different values of C (2,3,4,5) against the number of sweeps, for the first 1,000,000 iterations. In both the Figures we considered the

MTRJ-inv-II and GMTRJ-man-II algorithms with a number of trials $k = 10$.

Table 4: Posterior distribution of C for the latent class example with $k = 10$

C	RJ	MTRJ-inv-I	MTRJ-inv-II	GMTRJ-man-I	GMTRJ-man-II
1	0.000	0.000	0.000	0.000	0.000
2	0.210	0.211	0.211	0.213	0.213
3	0.218	0.217	0.213	0.215	0.225
4	0.174	0.180	0.182	0.178	0.176
5	0.131	0.130	0.134	0.131	0.131
6	0.092	0.094	0.093	0.091	0.090
7	0.063	0.063	0.061	0.060	0.060
8	0.040	0.040	0.040	0.040	0.039
9	0.026	0.025	0.025	0.026	0.024
10	0.017	0.016	0.016	0.017	0.016
$C \geq 11$	0.029	0.025	0.024	0.029	0.025

Table 5: Acceptance rate for the latent class example with $k = 10$

Move	RJ	MTRJ-inv-I	MTRJ-inv-II	GMTRJ-man-I	GMTRJ-man-II
split	1.99	5.89	7.24	5.84	4.03
combine	1.98	5.87	7.23	5.83	4.04
birth	5.34	11.11	11.00	8.88	8.64
death	5.35	11.08	10.95	8.87	8.63

Finally, in Table 6 are reported the values of the autocorrelation of the chain, for the most probable models (with a number of classes between 2 and 7), resulting in the five algorithms. We report the values corrected for the computing time.

From the results of the comparison, we can see that the MTRJ-inv and GMTRJ-man algorithms have a higher acceptance rate with respect to the RJ algorithm, lowering the autocorrelation of the chain. In particular, the GMTRJ-man-II algorithm, with $k = 10$, outperforms the other algorithms in terms of autocorrelation.

Moreover, although all the algorithms mix well over C , with few excursions in very high values of C , the mixing of the MTRJ-inv and GMTRJ-man algorithms is better than of the RJ algorithm (Figure 4). From Figure 5, we can see that for

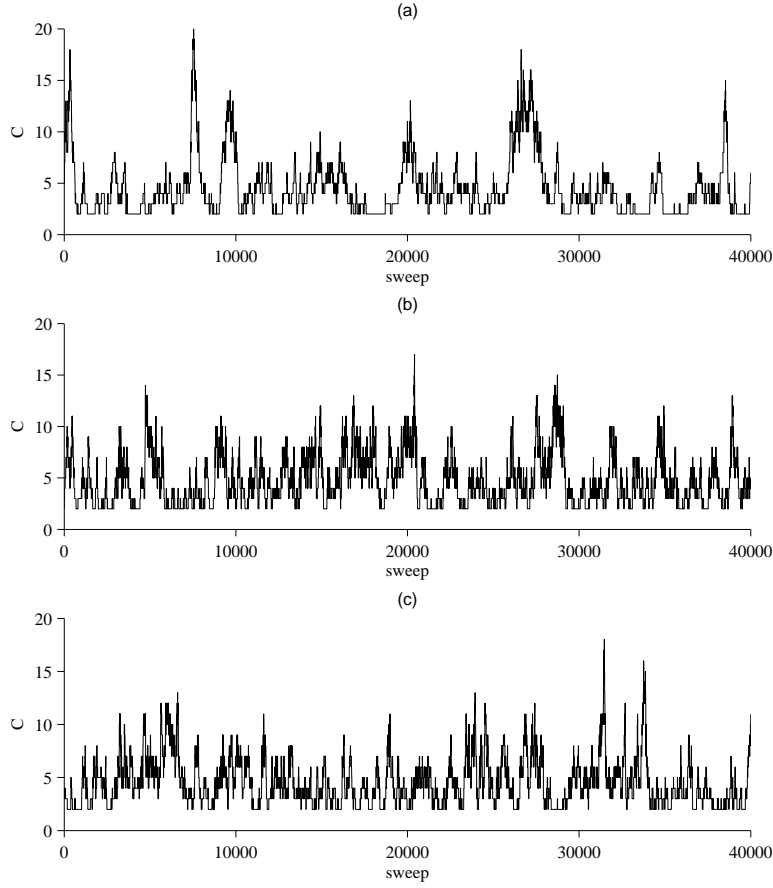


Figure 4: The number of latent classes in the first 40,000 iteration after the burn-in: (a) RJ, (b) MTRJ-inv-II with $k = 10$, (c) GMTRJ-man-II with $k = 10$

Table 6: Values (divided by 1,000 and adjusted for the computing time) of the auto-correlation of the chain for the latent class example

		Number of classes					
		2	3	4	5	6	7
	RJ	727.81	257.66	163.67	145.60	164.42	163.13
$k = 5$	MTRJ-inv-I	773.52	313.52	192.27	179.17	181.78	190.16
	MTRJ-inv-II	714.76	278.56	160.61	157.50	157.04	172.46
	GMTRJ-man-I	597.01	225.61	130.92	122.17	138.89	148.46
	GMTRJ-man-II	498.98	215.98	121.50	118.53	132.22	136.81
$k = 10$	MTRJ-inv-I	944.16	443.38	264.57	232.33	292.87	301.68
	MTRJ-inv-II	908.65	340.09	192.69	180.90	204.55	222.76
	GMTRJ	590.2	247.27	163.21	141.94	150.91	180.54
	GMTRJ-II	407.41	196.37	112.51	115.73	119.12	136.28

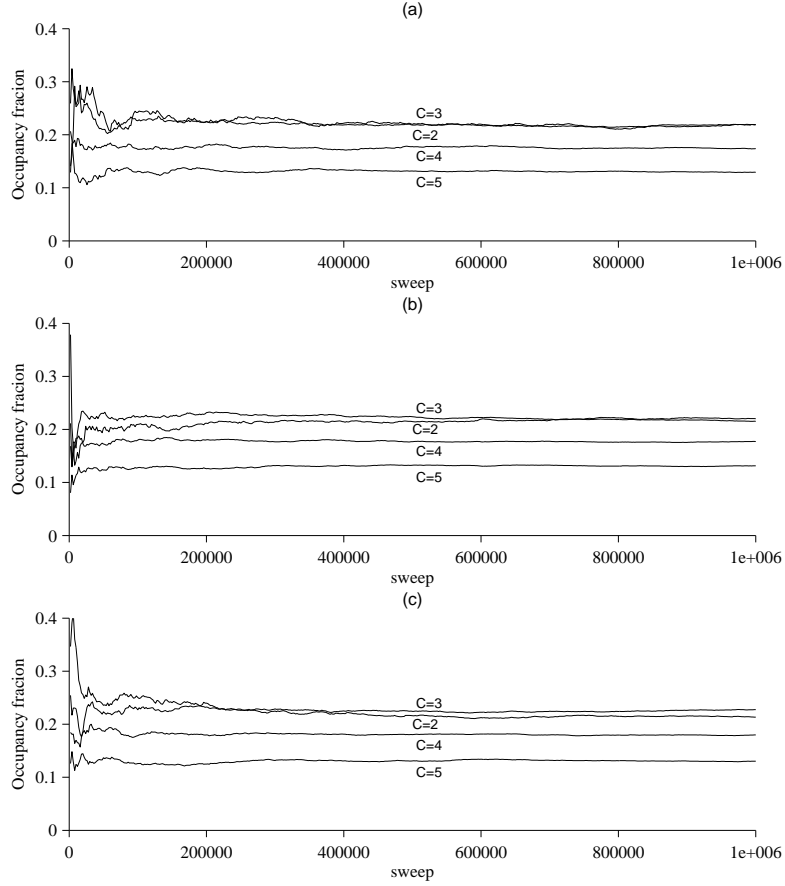


Figure 5: Occupancy fraction for different values of C : (a) RJ, (b) MTRJ-inv-II with $k = 10$, (c) GMTRJ-man-II with $k = 10$

all the algorithms the burn-in is more than adequate to achieve the stability in the occupancy fraction, but the proposed algorithms can achieve the stability in a shorter period.

5 Conclusions

In this paper we presented an extension of the RJ algorithm (GMTRJ algorithm) that allow us to explore the different models in a more efficient way, automating the choice of the jump proposal. The idea is to exploit the MTM paradigm in order to propose a fixed number of transdimensional moves, and then select one of them on the

basis of suitable selection probabilities. These probabilities are computed on the basis of a weighting function that could be arbitrarily chosen. The choice of this function is crucial for the effectiveness of the algorithm.

We illustrated several special cases of the algorithm coming out from this choice. Some of these algorithms may be seen as the corresponding versions, in Bayesian model choice context, of the MTM algorithm introduced by Liu et al. (2000) for Bayesian estimation problems; we termed these algorithms as MTRJ-I and MTRJ-inv. We also introduced alternative versions of the GMTRJ algorithm that could be useful in different model selection problems. The first version replaces the target distribution in the weighting function with its quadratic approximation so that the resulting algorithm, that we termed as GMTRJ-quad, is more effective than the standard RJ algorithm, without being more computationally intensive. We also demonstrated that, when for variable selection problem the computation of the quadratic approximation is not feasible, it is still possible to derive useful weighting functions that lead to an efficient algorithm.

We illustrated the potentialities of this approach by two real examples. The first was about the selection of covariates of a logistic regression model. For this case we compared the performance of the RJ algorithm with the performance of the MTRJ-I, MTRJ-inv, and GMTRJ-quad algorithms. We showed how, for different values of the parameters of the proposal distribution, the proposed algorithms always outperform the RJ algorithm. In particular, we have a lower autocorrelation of the chain and a higher acceptance rate. Moreover, the quadratic approximation allows us to obtain a gain of efficiency with respect to the other algorithms, when the computing time is properly taken into account.

The second example was about the estimation of the number of latent classes in a latent class model. This is an example in which the computation of the quadratic

approximation of the target distribution is not easy to derive. We therefore proposed to use the incomplete likelihood as weighting function; this choice allows us to save much computing time without loss of efficiency. The resulting version of the GMTRJ algorithm was named GMTRJ-man. The results obtained from applying the proposed approach to the latent class example confirm that, with the generalized MTM version of the RJ algorithm, is possible to increase the number of trials without increasing the computational effort, so as to better explore the state space.

Future research will be devoted to explore different types of the weighting function and to better evaluate how this choice affects the efficiency of the resulting algorithm. The aim is to automatize as much as possible the construction of the transdimensional moves.

References

- Al-Awadhi, F., Hurn, M., and Jennison, C. (2004). Improving the acceptance rate of reversible jump MCMC proposals. *Statistics & Probability Letters*, 69(2):189–198.
- Bartolucci, F., Scaccia, L., and Mira, A. (2006). Efficient bayes factor estimation from the reversible jump output. *Biometrika*, 93:41–52.
- Brooks, S., Giudici, P., and Roberts, G. O. (2003). Efficient construction of reversible jump Markov chain Monte Carlo proposal distributions. *Journal of the Royal Statistical Society, Series B*, 65:3–55.
- Cappe, O., Robert, C. P., and Ryden, T. (2003). Reversible jump, birth-and-death and more general continuous time Markov chain Monte Carlo samplers. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 65(3):679–700.

- Carlin, B. P. and Chib, S. (1995). Bayesian model choice via Markov chain Monte Carlo methods. *Journal for the Royal Statistical Society, Series B*, 57:473–484.
- Chib, S. (1995). Marginal likelihood from the Gibbs output. *Journal of the American Statistical Association*, 90:1313–1321.
- Chib, S. and Jeliazkov, I. (2001). Marginal likelihood from the Metropolis-Hastings output. *Journal of the American Statistical Association*, 96:270–281.
- Dellaportas, P., Forster, J., and Ntzoufras, I. (2002). On bayesian model and variable selection using MCMC. *Statistics and Computing*, 12:27–36.
- Fan, Y., Peters, G. W., and Sisson, S. A. (2009). Automating and evaluating reversible jump MCMC proposal distributions. *Statistics and Computing*, 19:409–421.
- Godsill, S. (2001). On the relationship between Markov chain Monte Carlo methods for model uncertainty. *Journal of computational and graphical statistics*, 10:230–248.
- Goodman, L. (1974). Exploratory latent structure analysis using both identifiable and unidentifiable models. *Biometrika*, 61(2):215–231.
- Green, P. and Mira, A. (2001). Delayed rejection in reversible jump Metropolis-Hastings. *Biometrika*, 88:1035–1053.
- Green, P. J. (1995). Reversible jump Markov chain Monte Carlo computation and bayesian model determination. *Biometrika*, 82:711–732.
- Green, P. J. (2003). Trans-dimensional Markov chain Monte Carlo. In Green, P., Hjort, N., and Richardson, S., editors, *Highly structured stochastic systems*, pages 179–198. Oxford: Oxford University Press.

- Green, P. J. and Hastie, D. (2009). Reversible jump MCMC. <http://www.maths.bris.ac.uk/~mapjg/Papers.html>.
- Han, C. and Carlin, B. P. (2000). Markov chain Monte Carlo methods for computing bayes factors: a comparative review. *Journal of the American Statistical Association*, 96:1122–1132.
- Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Journal of Chemical Physics*, 21:1087–1091.
- Liu, J. S. (2001). *Monte Carlo strategies in scientific computing*. Springer, New-York.
- Liu, J. S., Liang, F., and Wong, W. H. (2000). The Multiple-try Method and local optimization in Metropolis sampling. *Journal of American Statistical Association*, 95:121–134.
- Meng, X. L. and Wong, W. H. (1996). Simulating ratios of normalizing constants via a simple identity: a theoretical exploration. *Statistica Sinica*, 6:831–860.
- Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machine. *Journal of Chemical Physics*, 21:1087–1091.
- Pandolfi, S., Bartolucci, F., and Friel, N. (2010). A generalization of the Multiple-try Metropolis algorithm for bayesian estimation and model selection. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 581–588, Chia Laguna Resort, Sardinia, Italy.
- Richardson, S. and Green, P. (1997). On bayesian analysis of mixture with an unknown number of components. *Journal fo the Royal Statistical Society, Series B*, 59:731–792.

Robert, C. P. and Casella, G. (2004). *Monte Carlo statistical methods*. Springer, New-York.

Stephens, M. (2000). Bayesian analysis of mixture models with an unknown number of components - an alternative to reversible jump methods. *The Annals of Statistics*, 28:40–74.